

Package: geoelectrics (via r-universe)

September 22, 2026

Title 3D-Visualization of Geoelectric Resistivity Measurement Profiles

Version 0.2.2

Author Anja Kleebaum <kleebaum@informatik.uni-heidelberg.de>

Maintainer Anja Kleebaum <kleebaum@informatik.uni-heidelberg.de>

Description Visualizes two-dimensional geoelectric resistivity measurement profiles in three dimensions.

Depends R (>= 3.0), lattice, rgl, fields

Imports methods

License GPL (>= 2)

URL <https://github.com/kleebaum/geoelectrics>

BugReports <https://github.com/kleebaum/geoelectrics/issues>

Encoding UTF-8

Language en-US

RoxygenNote 6.1.0

Suggests testthat, tkrplot

Config/pak/sysreqs cmake libfreetype6-dev libglu1-mesa-dev make texlive libpng-dev libuv1-dev libgl1-mesa-dev zlib1g-dev

Repository <https://kleebaum.r-universe.dev>

Date/Publication 2023-02-01 21:09:38 UTC

RemoteUrl <https://github.com/kleebaum/geoelectrics>

RemoteRef HEAD

RemoteSha 7ca184a484e00f77d1b0183ff163ff8c52a2ae79

Contents

adjustHeight	2
calcRelativeCoords	3
geoelectricsGui	4
getHeightInformation	4

GpsCoordinates-class	5
levelplot	5
levelplotLegendLabel	7
myColorRamp	8
parseProcessedDataFile	8
parseRawDataFile	9
plot,Profile,ANY-method	10
plot3d,ProfileSet-method	11
plotIntersect	12
plotLegend	13
ProcessedData-class	15
Profile-class	15
ProfileSet-class	16
RawData-class	17
sinkhole	18
Index	19

<code>adjustHeight</code>	<i>Adjust Profile Height</i>
---------------------------	------------------------------

Description

Adjusts the height of a single profile (adds a delta value to ALL data points). This is necessary if GPS measurement heights of two profiles differ systematically.

Usage

```
adjustHeight(object, delta)

## S4 method for signature 'Profile'
adjustHeight(object, delta)
```

Arguments

- object a single Profile.
- delta positive or negative value.

Value

adjusted profile

See Also

[GpsCoordinates-class](#), [Profile-class](#)

Examples

```
p3 <- new(
  "Profile",
  title = "Profile 3",
  processedData =
    new("ProcessedData",
      address = system.file("extdata/processed/p3_DipolDipol_S-N.xyz",
        package='geoelectrics')),
  rawData =
    new("RawData",
      address = system.file("extdata/raw/p3_DipolDipol_S-N.dat",
        package='geoelectrics')),
  measurementType = "DipolDipol",
  gpsCoordinates =
    new("GpsCoordinates",
      address = system.file("extdata/gps/p3.txt",
        package='geoelectrics'))
)

p3 <- adjustHeight(p3, -10)
```

calcRelativeCoords	<i>Calculate Relative Coordinates</i>
--------------------	---------------------------------------

Description

Calculates relative coordinates (unity: meters) from GPS coordinates (either given in UTM or Gauss Krueger). This method is used when a profile set of many profiles is instantiated.

Usage

```
calcRelativeCoords(coords, minLat, minLon)
```

Arguments

coords	exact coordinates of a single Profile.
minLat	starting point (latitude).
minLon	starting point (longitude).

Value

data frame that contains the relative coordinates (latitude and longitude).

See Also

[ProfileSet-class](#), [GpsCoordinates-class](#)

geoelectricsGui *Graphical User Interface of the Geoelectrics R Package*

Description

This method starts a simple TclTk graphical user interface (GUI) provided by this package.

Usage

```
geoelectricsGui()
```

Examples

```
# geoelectricsGui()
```

getHeightInformation *Gets the Height Information for a Profile*

Description

Returns the heights for certain distances along the profile (topography information).

Usage

```
getHeightInformation(object)

## S4 method for signature 'ProcessedData'
getHeightInformation(object)
```

Arguments

object a single Profile.

Value

data frame containing distances and heights along the profile

See Also

[GpsCoordinates-class](#), [Profile-class](#), [ProcessedData-class](#)

Examples

```
data(sinkhole)

getHeightInformation(sinkhole@profiles[[1]]@processedData)
```

GpsCoordinates-class *GPS Coordinates Class*

Description

A class to handle gps coordinates.

Slots

address address of the gps ascii file
exact data frame that contains measured gps coordinates
relative relative coordinates, normalized to (0,0)
lm linear model of the measured gps coordinates
lmRelative linear model of relative coordinates

See Also

[Profile-class](#), [ProfileSet-class](#), [adjustHeight](#), [calcRelativeCoords](#)

Examples

```
gpsCoordinates = new('GpsCoordinates', address = system.file('extdata/gps/p1.txt',
  package='geoelectrics'))

data(sinkhole)
sinkhole@profiles[[1]]@gpsCoordinates
sinkhole@profiles[[1]]@gpsCoordinates@address
sinkhole@profiles[[1]]@gpsCoordinates@exact
sinkhole@profiles[[1]]@gpsCoordinates@lm
sinkhole@profiles[[1]]@gpsCoordinates@relative
sinkhole@profiles[[1]]@gpsCoordinates@lmRelative
```

levelplot *Levelplot of Geoelectrics Data*

Description

Plots the interpolated resistance values of the geoelectrics data.

Usage

```

levelplot(x, data, ...)

## S4 method for signature 'Profile'
levelplot(x, dataType = "processed",
  withTopo = FALSE, xlab = "Length [m]", ylab = "Depth [m]",
  main = paste(x@title), col = colors, breaks = 18, trafo = log,
  backtrafo = exp, aspect = "iso", ...)

## S4 method for signature 'ProfileSet'
levelplot(x, dataType = "processed",
  withTopo = FALSE, xlab = "Length [m]", ylab = "Depth [m]",
  main = paste(x@title), col = colors, breaks = 18, trafo = log,
  backtrafo = exp, aspect = "iso", ...)

levelplotProcessedData(x, xlab = "Length [m]", ylab = "Depth [m]",
  main = paste(x@title, "without topography"), col = colors,
  breaks = 18, trafo = log, backtrafo = exp, aspect = "iso", ...)

levelplotProcessedDataWithTopo(x, xlab = "Length [m]",
  ylab = "Height [m]", main = paste(x@title, "with topography"),
  col = colors, breaks = 18, trafo = log, backtrafo = exp,
  aspect = "iso", ...)

levelplotRawData(x, xlab = "Length [m]", ylab = "Depth [m]",
  main = paste(x@title, "without topography (raw data)"), col = colors,
  trafo = log, aspect = "iso", ...)

```

Arguments

<code>x</code>	profile object.
<code>data</code>	is always NULL
<code>...</code>	lattice levelplot arguments.
<code>dataType</code>	specify whether 'processed' (default) or 'raw' data should be plotted
<code>withTopo</code>	TRUE if topography information is plotted
<code>xlab</code>	label for x-axes.
<code>ylab</code>	label for y-axes.
<code>main</code>	title to be plotted.
<code>col</code>	vector of colors.
<code>breaks</code>	number of color breaks.
<code>trafo</code>	transformation to be done on data (default: log).
<code>backtrafo</code>	back transformation to plot correct labels (default: exp).
<code>aspect</code>	the y/x aspect ratio (default: iso).

See Also[Profile-class](#)**Examples**

```
data(sinkhole)

levelplot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = FALSE)
levelplotLegendLabel()

levelplot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = TRUE)
levelplotLegendLabel()

levelplot(sinkhole@profiles[[1]], dataType = 'raw')
levelplotLegendLabel()
```

levelplotLegendLabel	<i>Levelplot Legend Label</i>
----------------------	-------------------------------

Description

Plots the label of the levelplot.

Usage

```
levelplotLegendLabel(legend.lab = "Resistivity",
  unit = expression(paste("[", Omega, "m]")))
```

Arguments

legend.lab	label (default: 'Resistivity').
unit	unit (default: 'Ohm*m').

See Also[levelplot](#)**Examples**

```
data(sinkhole)

levelplot(sinkhole@profiles[[1]])
levelplotLegendLabel()

levelplot(sinkhole@profiles[[2]])
levelplotLegendLabel()

levelplot(sinkhole@profiles[[3]])
levelplotLegendLabel()
```

myColorRamp	<i>Maps color to resistivity value</i>
-------------	--

Description

Maps color to (resistivity) values. A minimum and maximum value can be specified.

Usage

```
myColorRamp(col, values, minData = min(values), maxData = max(values))
```

Arguments

col	Character vector of colors.
values	Numeric vector of values.
minData	Minimum value (default min(values)). All smaller values will assigned to the first color in vector col.
maxData	Maximum value (default max(values)). All higher values will assigned to the last color in vector col.

parseProcessedDataFile	<i>Parses a Processed Data File</i>
------------------------	-------------------------------------

Description

Parses .xyz files produced by the software Res2DInv. Needs to be overwritten if another processed data format is used.

Usage

```
parseProcessedDataFile(address, skip = 0)
```

Arguments

address	address of the raw data ascii file.
skip	the number of lines of the data file to skip before beginning to read data.

Value

list of two data frames: The first data frame contains points without topography (distance, depth and resistivity values). The second data frame contains points with topography (distance, height and resistivity values).

See Also

[ProcessedData-class](#), [Profile-class](#)

Examples

```
fileAddress <- system.file('extdata/processed/p1_DipolDipol_SW-NE.xyz',
                           package = 'gEOELECTRICS')

processedData = new('ProcessedData')
processedData@address = fileAddress
processedData@points <- parseProcessedDataFile(address = fileAddress)[[1]]
processedData@pointsWithTopo <- parseProcessedDataFile(address = fileAddress)[[2]]
```

parseRawDataFile	<i>Parses a Raw Data File</i>
------------------	-------------------------------

Description

Parses a geoelectrics raw data file created by the GeoTest software by Dr. Rauen. Needs to be overwritten if another raw data format is used.

Usage

```
parseRawDataFile(address, skip = 9)
```

Arguments

address	address of the raw data ascii file.
skip	the number of lines of the data file to skip before beginning to read data.

Value

data frame containing distance, depth and resistivity values

See Also

[RawData-class](#), [Profile-class](#)

Examples

```
fileAddress <- system.file('extdata/raw/p1_DipolDipol_SW-NE.dat',
                           package = 'gEOELECTRICS')

rawData = new('RawData')
rawData@address = fileAddress
rawData@points <- parseRawDataFile(address = fileAddress)
```

plot,Profile,ANY-method

Plot Geoelectrics Data Points

Description

Plots the geoelectrics data points of a profile.

Usage

```
## S4 method for signature 'Profile,ANY'
plot(x, dataType = "processed", withTopo = T,
     xlab = "Length [m]", ylab = "Height [m]", main = paste(x@title,
     "with topography"), asp = 1, ...)

## S4 method for signature 'ProfileSet,ANY'
plot(x, dataType = "processed",
     withTopo = T, xlab = "Length [m]", ylab = "Height [m]",
     main = paste(x@title, "with topography"), asp = 1, ...)

plotProcessedData(x, xlab = "Length [m]", ylab = "Depth [m]",
  main = paste(x@title, "without topography"), ...)

plotProcessedDataWithTopo(x, xlab = "Length [m]", ylab = "Height [m]",
  main = paste(x@title, "with topography"), ...)

plotRawData(x, xlab = "Length [m]", ylab = "Depth [m]",
  main = paste(x@title, "without topography"), ...)

plotRawDataWithTopo(x, xlab = "Length [m]", ylab = "Depth [m]",
  main = paste(x@title, "with topography"),
  height = x@processedData@height, spline = TRUE, ...)
```

Arguments

x	profile object.
dataType	specify whether 'processed' (default) or 'raw' data should be plotted
withTopo	TRUE if topography information is plotted
xlab	label for x-axes.
ylab	label for y-axes.
main	title to be plotted.
asp	the y/x aspect ratio (default: 1).
...	plot parameters (such as pch, cex, col, ...).
height	topo data frame of distances and height.
spline	if TRUE spline interpolation is conducted.

See Also

[Profile-class](#), [plot3d](#), [levelplot](#)

Examples

```
data(sinkhole)

plot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = FALSE)
plotProcessedData(sinkhole@profiles[[1]])

plot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = TRUE)
plotProcessedDataWithTopo(sinkhole@profiles[[1]])

plot(sinkhole@profiles[[1]], dataType = 'raw', withTopo = FALSE)
plotRawData(sinkhole@profiles[[1]])

plot(sinkhole@profiles[[1]], dataType = 'raw', withTopo = TRUE)
plotRawDataWithTopo(sinkhole@profiles[[1]])
```

plot3d,ProfileSet-method

3D Scatterplot of Geoelectrics Profiles

Description

Plots the interpolated resistance values of the processed data for a single profile or a set of profiles.

Usage

```
## S4 method for signature 'ProfileSet'
plot3d(x, title = x@title, sub = "",
       xlab = "", ylab = "", zlab = "", minData = x@minData,
       maxData = x@maxData, col = colors, trafo = log,
       psize = pointsize, ...)

## S4 method for signature 'Profile'
plot3d(x, title = "", sub = "", xlab = "",
       ylab = "", zlab = "", minData = x@processedData@minData,
       maxData = x@processedData@maxData, col = colors, trafo = log,
       psize = pointsize, ...)
```

Arguments

x	either an object of a single Profile or a ProfileSet.
title	title to be plotted.
sub	subtitle to be plotted.
xlab	label of the x-axes, e.g. length [m].

ylab	label of the y-axes, e.g. height above sea level [m].
zlab	label of the z-axes, e.g. length [m].
minData	mimimum value to adjust color bar.
maxData	maximum value to adjust color bar.
col	vector of colors.
trafo	transformation to be done on data (default: log).
psize	size of value points (default: 10).
...	parameters passed to points3d method of rgl package

See Also

[Profile-class](#), [ProfileSet-class](#), [plot](#), [levelplot](#)

Examples

```
data(sinkhole)

plot3d(sinkhole@profiles[[1]])
plot3d(sinkhole)
```

plotIntersect

Plot Profile Intersection

Description

Plots resistivity against height on and next to the intersection line between two profiles.

Usage

```
plotIntersect(.Object1, .Object2 = NULL,
  xlab = "Height above sea level [m]",
  ylab = expression(paste("Resistivity [", Omega, "m]")), main = "",
  trafo = log, backtrafo = exp, col = colors, pch = c(20, 20),
  type = "p", legendLoc = "bottomleft")

## S4 method for signature 'ProfileSet,ANY'
plotIntersect(.Object1, xlab, ylab, main, trafo,
  backtrafo, col, pch, type, legendLoc)

## S4 method for signature 'Profile,Profile'
plotIntersect(.Object1, .Object2 = NULL,
  xlab = "Height above sea level [m]",
  ylab = expression(paste("Resistivity [", Omega, "m]")), main = "",
  trafo = log, backtrafo = exp, col = colors, pch = c(20, 20),
  type = "p", legendLoc = "bottomleft")
```

Arguments

.Object1	either a single Profile or a ProfileSet.
.Object2	either a second single Profile or NULL if .Object1 is of type ProfileSet.
xlab	label of the x-axes, e.g. length [m].
ylab	label of the y-axes, e.g. height above sea level [m].
main	title to be plotted.
trafo	transformation to be done on data (default: log).
backtrafo	back transformation to plot correct labels (default: exp).
col	character vector of colors.
pch	numeric vector of plotting symbols.
type	plot type (default "p" for points). "b" for both points and lines, "c" for empty points joined by lines, "o" for overplotted points and lines, "s" and "S" for stair steps and "h" for histogram-like vertical lines. Finally, "n" does not produce any points or lines.
legendLoc	legendLocation (default "bottomleft").

See Also

[ProfileSet-class](#)

Examples

```
data(sinkhole)

plotIntersect(sinkhole)
plotIntersect(sinkhole@profiles[[1]], sinkhole@profiles[[2]])
```

plotLegend

Plots Legend

Description

Plots the legend for resistivity values.

Usage

```
plotLegend(.Object, legend.lab = expression(paste("Resistivity [", Omega,
  " m]")), minData = 0, maxData = 999999, breaks = NULL,
  legend.line = 2.2, nlevel = 18, lab.breaks = c(), horizontal = T,
  col = colors, trafo = log, backtrafo = exp, ...)

## S4 method for signature 'ProfileSet'
plotLegend(.Object, legend.lab,
  minData = .Object@minData, maxData = .Object@maxData)
```

```
## S4 method for signature 'Profile'
plotLegend(.Object, legend.lab,
  minData = .Object@processedData@minData,
  maxData = .Object@processedData@maxData)
```

Arguments

<code>.Object</code>	either a single Profile or a ProfileSet.
<code>legend.lab</code>	label of legend (default: <code>expression(paste("Resistivity [", Omega, "]"))</code>).
<code>minData</code>	minimum value.
<code>maxData</code>	maximum value.
<code>breaks</code>	Break points in sorted order to indicate the intervals for assigning the colors. Note that if there are <code>nlevel</code> colors there should be <code>(nlevel+1)</code> breakpoints. If <code>breaks</code> is not specified <code>(nlevel+1)</code> equally spaced breaks are created where the first and last bin have their midpoints at the minimum and maximum values in <code>z</code> or at <code>zlim</code> .
<code>legend.line</code>	distance in units of character height (as in <code>mtext</code>) of the legend label from the color bar. Make this larger if the label collides with the color axis labels.
<code>nlevel</code>	number of color levels.
<code>lab.breaks</code>	number of breaks.
<code>horizontal</code>	If false legend will be a vertical strip on the right side. If true (default) the legend strip will be along the bottom.
<code>col</code>	vector of colors.
<code>trafo</code>	transformation to be done on data (default: <code>log</code>). For linear scale: <code>function(x) x</code> .
<code>backtrafo</code>	back transformation to plot correct labels (default: <code>exp</code>). For linear scale: <code>function(x) x</code> .
<code>...</code>	<code>image.plot</code> arguments.

See Also

[Profile-class](#), [ProfileSet-class](#), [plot3d](#),

Examples

```
data(sinkhole)

plotLegend(sinkhole)

# for linear scale:
plotLegend(sinkhole@profiles[[1]],
  trafo=function(x) x,
  backtrafo=function(x) x,
  minData=100, maxData=50000)
```

ProcessedData-class	<i>Processed Data Class</i>
---------------------	-----------------------------

Description

A class to handle processed geoelectrics data in ascii format. The processed data class parses .xyz files produced by the software Res2DInv. If you want to use another format, overwrite the [parseProcessedDataFile](#) method.

Slots

address address of the processed ascii file
 points data frame that contains positions and values without topography information
 pointsWithTopo data frame that contains positions and values with topography information
 height data frame that contains topography information (distances and heights). It is reconstructed from .xyz-file.
 minData minimum value
 maxData maximum value

See Also

[parseProcessedDataFile](#), [Profile-class](#), [ProfileSet-class](#)

Examples

```
processedData = new('ProcessedData',
  address = system.file('extdata/processed/p1_DipolDipol_SW-NE.xyz',
    package='geoelectrics'))

data(sinkhole)
sinkhole@profiles[[1]]@processedData
sinkhole@profiles[[1]]@processedData@points
sinkhole@profiles[[1]]@processedData@pointsWithTopo
sinkhole@profiles[[1]]@processedData@height
sinkhole@profiles[[1]]@processedData@minData
sinkhole@profiles[[1]]@processedData@maxData
```

Profile-class	<i>Profile Class</i>
---------------	----------------------

Description

A class to handle a single profile.

Slots

title title of the profile (e.g. Profile 1).

number index of the profile.

processedData object of Processed Data Class ([ProcessedData-class](#)).

rawData object of Raw Data Class ([RawData-class](#)).

measurementType type of measurement (e.g. Dipole Dipole, Wenner, ...).

gpsCoordinates object of GpsCoordinates Class ([GpsCoordinates-class](#)).

See Also

[ProcessedData-class](#), [RawData-class](#), [GpsCoordinates-class](#), [plot3d](#), [plot](#)

Examples

```
p1 <- new('Profile',
  title = 'Profile 1',
  processedData =
    new('ProcessedData', address = system.file('extdata/processed/p1_DipolDipol_SW-NE.xyz',
      package='gEOELECTRICS')),
  rawData =
    new('RawData', address = system.file('extdata/raw/p1_DipolDipol_SW-NE.dat',
      package='gEOELECTRICS')),
  measurementType = 'DipoleDipole',
  gpsCoordinates =
    new('GpsCoordinates', address = system.file('extdata/gps/p1.txt',
      package='gEOELECTRICS'))

p1@title
p1@processedData
p1@rawData
p1@measurementType
p1@gpsCoordinates

plot3d(p1)
```

ProfileSet-class

Profile Set Class

Description

A class to handle a collection of many profiles.

Slots

title title to plot
 profiles list that contains objects of class Profile ([Profile-class](#))
 minLat minimum latitude value of all profiles
 minLon minimum longitude value of all profiles
 minData minimum data value of all profiles
 maxData maximum data value of all profiles

See Also

[Profile-class](#), [plot3d](#)

Examples

```

# sinkhole <- new('ProfileSet',
#               profiles = list(p1, p2, p3),
#               title='Sinkhole')

data(sinkhole)
plot3d(sinkhole)

```

RawData-class

Raw Data Class

Description

A class to handle geoelectrics raw data. The raw data class parses .dat files provided by the GeoTest software by Dr. Rauen. If you want to use another format, overwrite the [parseRawDataFile](#) method.

Slots

address address of the raw data ascii file.
 points data frame that contains raw data resistance values and their positions (distance and depth).

See Also

[parseRawDataFile](#), [Profile-class](#), [ProfileSet-class](#)

Examples

```

rawData = new('RawData', address = system.file('extdata/raw/p1_DipolDipol_SW-NE.dat',
        package='geoelectrics'))

data(sinkhole)
sinkhole@profiles[[2]]@rawData
sinkhole@profiles[[2]]@rawData@address
sinkhole@profiles[[2]]@rawData@points

```

sinkhole

*Filled Sinkhole***Description**

Geoelectrics profiles measured at a filled sinkhole. This data set contains an object of the ProfileSet class.

Format

Object of Profile Set class including three Profiles.

See Also

[ProfileSet-class](#), [Profile-class](#), [plot3d](#), [plot](#), [levelplot](#), [plotLegend](#), [plotIntersect](#)

Examples

```
data(sinkhole)

# Plot the processed data in three dimensions
plot3d(sinkhole)

# Plot legend for the profile set
plot.new()
plotLegend(sinkhole)

# Plot raw data of profile 1
plot(sinkhole@profiles[[1]], dataType = 'raw', withTopo = FALSE,
     main = 'Profile 1', ylab = 'Depth [m]')

plot(sinkhole@profiles[[1]], dataType = 'raw', withTopo = TRUE)

levelplot(sinkhole@profiles[[1]], dataType = 'raw')
levelplotLegendLabel()

# Plot processed data of profile 1
plot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = FALSE,
     main = 'Profile 1', ylab = 'Depth [m]')

plot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = TRUE)

levelplot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = FALSE)
levelplotLegendLabel()

levelplot(sinkhole@profiles[[1]], dataType = 'processed', withTopo = TRUE)
levelplotLegendLabel()

# Compare processed data values on the intersection line between two profiles
plotIntersect(sinkhole)
```

Index

- * **datasets**
 - sinkhole, [18](#)
- adjustHeight, [2](#), [5](#)
- adjustHeight, Profile-method
 - (adjustHeight), [2](#)
- calcRelativeCoords, [3](#), [5](#)
- geoelectricsGui, [4](#)
- getHeightInformation, [4](#)
- getHeightInformation, ProcessedData-method
 - (getHeightInformation), [4](#)
- GpsCoordinates-class, [5](#)
- levelplot, [5](#), [7](#), [11](#), [12](#), [18](#)
- levelplot, Profile-method (levelplot), [5](#)
- levelplot, ProfileSet-method
 - (levelplot), [5](#)
- levelplotLegendLabel, [7](#)
- levelplotProcessedData (levelplot), [5](#)
- levelplotProcessedDataWithTopo
 - (levelplot), [5](#)
- levelplotRawData (levelplot), [5](#)
- myColorRamp, [8](#)
- parseProcessedDataFile, [8](#), [15](#)
- parseRawDataFile, [9](#), [17](#)
- plot, [12](#), [16](#), [18](#)
- plot (plot, Profile, ANY-method), [10](#)
- plot, Profile, ANY-method, [10](#)
- plot, ProfileSet, ANY-method
 - (plot, Profile, ANY-method), [10](#)
- plot3d, [11](#), [14](#), [16–18](#)
- plot3d (plot3d, ProfileSet-method), [11](#)
- plot3d, Profile-method
 - (plot3d, ProfileSet-method), [11](#)
- plot3d, ProfileSet-method, [11](#)
- plotIntersect, [12](#), [18](#)
- plotIntersect, Profile, Profile-method
 - (plotIntersect), [12](#)
- plotIntersect, ProfileSet, ANY-method
 - (plotIntersect), [12](#)
- plotLegend, [13](#), [18](#)
- plotLegend, Profile-method (plotLegend), [13](#)
- plotLegend, ProfileSet-method
 - (plotLegend), [13](#)
- plotProcessedData
 - (plot, Profile, ANY-method), [10](#)
- plotProcessedDataWithTopo
 - (plot, Profile, ANY-method), [10](#)
- plotRawData (plot, Profile, ANY-method), [10](#)
- plotRawDataWithTopo
 - (plot, Profile, ANY-method), [10](#)
- ProcessedData-class, [15](#)
- Profile-class, [15](#)
- ProfileSet-class, [16](#)
- RawData-class, [17](#)
- sinkhole, [18](#)